

Memory Based Collaborative Filtering with Lucene

Claudio Gennaro
claudio.gennaro@isti.cnr.it
ISTI-CNR, Pisa, Italy

July 4, 2016

Abstract

Memory Based Collaborative Filtering is a widely used approach to provide recommendations. It exploits similarities between ratings across a population of users by forming a weighted vote to predict unobserved ratings. Bespoke solutions are frequently adopted to deal with the problem of high quality recommendations on large data sets. A disadvantage of this approach, however, is the loss of generality and flexibility of the general collaborative filtering systems. In this paper, we have developed a methodology that allows one to build a scalable and effective collaborative filtering system on top of a conventional full-text search engine such as Apache Lucene.

1 Introduction

Collaborative Filtering (CF) is the task of predicting unknown rates of a given user for some items (e.g., products, movies) by taking into account other users' preferences. One of the most commonly employed mechanism to approach the problem of CF, often referred to as memory-based CF, it is to compute similarity between users or items on the entire collection of previously rated items by the users [8]. Depending on how the neighborhood is defined, memory based CF comes in two flavors: user-based or item-based CF. Although, the term “collaborative filtering” is commonly used as a synonym for user-based CF [12], the two approaches share the same common abstract formulation: both deal with the prediction of unassigned elements of a given matrix relying on the statistics of elements already assigned. Rows and columns take the role of users and items, while matrix's elements represent rates given by the users on items. Within this abstract view, memory-based CF exploits similarities across a population of vectors from the matrix above by forming weighted rates to predict unrated items.

This observation provides the basis for a link between CF and information retrieval (IR). For instance, user-based CF relies on the calculation of the

similarity between users almost in the same way as IR systems calculate the similarity between documents. Both their algorithms essentially calculate the dot product between vectors, which take the form of user preferences, in the case of CF, and documents, in the case of IR.

The work by Breese et al. [2] is definitely the first research that has found connections between CF and IR. It used the cosine similarity in a vector space model, in which users take the role of documents, titles take the role of words, and rates take the role of word frequencies. Although, this first attempt is only valid for the user-based CF approach, it can be easily extended to the case of item-based CF by replacing the users with the items. In [4], the authors propose the use of an inverted index (the core of any IR system) for user-based CF in order to speed up the execution time. However, they had to build a CF system from scratch, which is in general challenging and resource demanding.

This fact suggests that the idea of building a CF on top of a conventional IR system is not so absurd. Unfortunately, in spite of the similarity between the two approaches, there are essential differences in the way in which the weights of the vectors are eventually calculated. In fact, IR systems typically use the well-known tf-idf weighing scheme. Moreover, in memory based CF, one difficulty in measuring the user-user similarity is that the raw ratings may contain biases caused by the different behaviors of different users. For example, some users may tend to give high ratings [6]. To correct such biases, one solution is to “mean-center” the user rates before computing the cosine similarity, which corresponds to consider the Pearson correlation.

To this end, we propose an approach to CF, which uses a document-based representation of users/items and their rates so that when documents are indexed by a conventional retrieval system, such as Lucene (<http://lucene.apache.org>), the vector weights assume the form of mean-centered rates.

A similar approach to the one we propose can be found in [1]. However, this approach to CF has the disadvantage of using a cumbersome transformation technique in which for each item i (represented as document) they store in an inverted index the items (now acting as terms) most similar to i . Once the inverted index has been built, a conventional text retrieval engine needs two more operations (called query transformation and processing) in order to generate a document ranking for a particular query.

As it will be shown, our approach to CF is much more practicable and flexible. We exploit Lucene features to build an index from the free movie recommender site MovieLens in which documents representing user profiles are stored. An experimental evaluation shows that our method produces results that are even better than those that would obtain using the state-of-art techniques for CF.

2 Memory-Based CF

Let U be a set of N users, I a set of M items, and let $r_{u,i}$ denote the rating of users $u \in U$ on item $i \in I$. $S_u \subseteq I$ stands for the set of items that user u

has rated and $S_i \subseteq U$ is the set of users that have rated item i . The purpose of CF is to predict the rating $p_{u,i}$ of a given user u on a given item i . User u is supposed to be active, meaning that he/she has already rated some items, therefore $S_u \neq \emptyset$. But the item to be predicted shall not be known by the user, so $i \notin S_u$ [3].

In user-based CF, we compute similarity between users. To accomplish this task the k -Nearest-Neighbors (kNN) of a target user's profile with the historical profiles of other users are retrieved. The recommendation is then made by aggregating the ratings of these neighbors. In item-based CF, we similarly look for items sharing common user preferences. Therefore, in this case, we compute the kNN of a target item.

2.1 Similarity Functions

Several similarity functions have been proposed for measuring both the similarity between users or items to find the kNN. These functions are sometimes referred to as *correlation coefficients*. Probably the most popular and also the oldest one is the Pearson correlation introduced in [8]. It corresponds to the cosine of rates' deviations from the mean, which for users-based approach is given by:

$$sim_P(u, a) = \frac{\sum_{i \in S_u \cap S_a} (r_{u,i} - \bar{r}_{u,a})(r_{a,i} - \bar{r}_{a,u})}{\sqrt{\sum_{i \in S_u \cap S_a} (r_{u,i} - \bar{r}_{u,a})^2} \sqrt{\sum_{i \in S_u \cap S_a} (r_{a,i} - \bar{r}_{a,u})^2}}$$

where $\bar{r}_{u,a} = (\sum_{i \in S_u \cap S_a} r_{u,i}) / |S_u \cap S_a|$. While, in the item-based approach, we have:

$$sim_P(i, j) = \frac{\sum_{a \in S_i \cap S_j} (r_{a,i} - \bar{r}_{i,j})(r_{a,j} - \bar{r}_{j,i})}{\sqrt{\sum_{a \in S_i \cap S_j} (r_{a,i} - \bar{r}_{i,j})^2} \sqrt{\sum_{a \in S_i \cap S_j} (r_{a,j} - \bar{r}_{j,i})^2}}$$

where $\bar{r}_{i,j} = (\sum_{a \in S_i \cap S_j} r_{a,i}) / |S_i \cap S_j|$. Another popular similarity function is the adjusted cosine similarity (ACS), sim_A , introduced by [10]. It corresponds to the cosine of the angle between the user/item rating vectors, after adjusting each rating by subtracting their average rating (mean-centered rates). For user-based approach sim_A is defined as:

$$sim_A(u, a) = \frac{\sum_{i \in S_u \cap S_a} (r_{u,i} - \bar{r}_u)(r_{a,i} - \bar{r}_a)}{\sqrt{\sum_{i \in S_u \cap S_a} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in S_u \cap S_a} (r_{a,i} - \bar{r}_a)^2}} \quad (1)$$

where $\bar{r}_u = (\sum_{i \in S_u} r_{u,i}) / |S_u|$. While, in the item-based approach, we have:

$$sim_A(i, j) = \frac{\sum_{a \in S_i \cap S_j} (r_{a,i} - \bar{r}_i)(r_{a,j} - \bar{r}_j)}{\sqrt{\sum_{a \in S_i \cap S_j} (r_{a,i} - \bar{r}_i)^2} \sqrt{\sum_{a \in S_i \cap S_j} (r_{a,j} - \bar{r}_j)^2}} \quad (2)$$

where

$$\bar{r}_i = (\sum_{a \in S_i} r_{a,i}) / |S_i|.$$

2.2 Predicting Unrated Items

Now that we have defined the notion of similarity between users and items, we show how to predict the rating of an unrated item. In memory-based approach to CF, the similarity function can be seen as “plug-in” component of the filtering algorithm as discussed above. For predicting the rating of user a on item j , user-based CF uses the average rating \bar{r}_a , and the set of kNN (\mathcal{N}_a) to user a as follows:

$$p_{a,j} = \bar{r}_a + \frac{\sum_{u \in \mathcal{N}_a} \text{sim}(a, u) \times (r_{u,j} - \bar{r}_u)}{\sum_{u \in \mathcal{N}_a} |\text{sim}(a, u)|} \quad (3)$$

In general this type of estimate can be thought of as the sum of two contributions called *base estimate* and *neighborhood estimate* [7]. Where the former contribution can be taken as a minimal estimate.

Item-based CF is the dual approach to the user-based approach and uses the similarities between items rather than similarities between users. As it is easy to see, we can use again a technique based on kNN (\mathcal{M}_i) of item i just as we did for the user-based approach. We predict the rating of user a on item i as follows:

$$p_{a,i} = \bar{r}_i + \frac{\sum_{j \in \mathcal{M}_i} \text{sim}(i, j) \times (r_{a,j} - \bar{r}_j)}{\sum_{j \in \mathcal{M}_i} |\text{sim}(i, j)|} \quad (4)$$

In this paper, we focus on the ACS, since it is shown to be the most accurate similarity measure for memory-based CF systems [10]. However, the main reason why we use ACS is related to the way in which the means are evaluated and to the idea of using a conventional text search engine as the basic tool to support CF. Let us take the user-based approach as example. Both Pearson and ACS for two users u and a are computed just over the ratings in $S_u \cap S_a$ because we want to consider only the co-rated items of the users u and a . However, the “global” average rating \bar{r}_u and \bar{r}_a of the adjusted cosine are computed considering all the items rated by u and a . While, the mean $\bar{r}_{u,a}$ in Pearson correlation is the average rating of user u for all the items in $S_u \cap S_a$. Therefore, in general $\bar{r}_{u,a} \neq \bar{r}_{a,u}$. The advantage of using the global average rating \bar{r}_u is that it can be easily evaluated and stored at indexing time since it refers to a single user only. The average rate $\bar{r}_{a,u}$, used in Pearson correlation, should be evaluated at run-time, since it considers only rates of user a on items that are in $S_a \cap S_u$. The same reasoning applies to item-based case.

3 CF with Lucene

Let us now enter the core of our method. In what follows, our aim will be to employ a text retrieval, such as Lucene, for easily developing a memory-based CF framework. To this end, in case of user-based CF, we can adopt a transformation in which users take the role of documents, items take the role of words, and rates take the role of word frequencies [2]. Similarly, for item-based CF, we could let users take the role of words, items take the role of documents, and rates take the role of word frequencies. Since in standard text retrieval the similarity between two documents is often measured by treating each document as a vector of word frequencies and computing the cosine similarity by the two

vectors, we can exploit the above transformation to compute similarity between users or items. This basic approach would allow us to retrieve kNN documents (i.e., users or items) on the basis of their cosine similarity (sim_C) as follows

$$sim_C(a, u) = \frac{\sum_{i \in S_a \cap S_u} f_{a,i} f_{u,i}}{\sqrt{\sum_{i \in S_a \cap S_u} f_{a,i}^2} \sqrt{\sum_{i \in S_a \cap S_u} f_{u,i}^2}}, \quad (5)$$

where $f_{u,i}$ now represents the frequency of word i in document u (which in this case is equal to $r_{u,i}$). An analogous formula can be written for the item-based case between two items i and j . However, as proved in [10] ACS performs best among other similarities in the CF by eliminating biases caused by the different rating behaviors of different users. Our goal therefore is to obtain the ACS on top of the text retrieval library Lucene. To this end, for user-based CF, a first approach is to make the frequency of word i in the document u proportional to its mean-centered rates, i.e. the term $r_{u,i} - \bar{r}_u$. For the item-based CF, the document i represents an item, therefore the frequency of a word u will be proportional the term $r_{u,i} - \bar{r}_i$. We call the terms $r_{u,i} - \bar{r}_u$ and $r_{u,i} - \bar{r}_i$ Mean-Centered Rates (MCRs). Word frequencies are proportional to MCRs because generally they are decimal numbers, but frequencies are integers. Therefore, we can multiply all the MCRs by a common integer factor (say 100).

For example, in case of user-based CF, if the rate given by the user u to the item with id 37 is 4, and the average value for the user is 3.95, then the MCR will be $4 - 3.95 = 0.05$. We set the frequency of the word corresponding to that user to $0.05 \times 100 = 5$. Note that by multiplying all the MCRs for the same factor 100, we do not affect the efficacy of the ACS because the frequencies are eventually normalized (apart from the truncation to two decimal places). Finally, the document corresponding to user u will contain the movie id 37 repeated five times, i.e., “37 37 37 37 37”. It is easy to see that by substituting the words frequencies (which are proportional to the MCRs) in place of the user rates in Eq. (5), we obtain the formula of the ACS.

However, we still have to face another problem: how to deal with negative values of MCRs. To this end, we group the MCRs of a document into two categories: positive and negative MCRs. We refer the former group and the latter group as pMCRs and nMCRs, respectively. Positive contributions to the numerator of the ACS in Eq. (1) are given by the product of pMCRs with themselves plus the product of the nMCRs with themselves. Therefore, if we associate separate ids to the pMCRs and nMCRs, these will contribute to the cosine similarity independently, exactly as described in Eq. (1). Continuing with our example of user-based CF, suppose we have another item of id, (say 24) corresponding to a nMCRs = -0.04. We decide then to use the prefix ‘p’ for the id corresponding to the pMCRs while we use the prefix ‘n’ for the nMCRs. The content of the document (i.e., the user u) therefore will be “p37 p37 p37 p37 p37 n24 n24 n24 n24”. Suppose we have the following query, “p37 p37 p37 p24 p24 p24 p24 p24”, which corresponds to a query user a of which we want to calculate the similarity with the above user. The matching function (standard cosine similarity) of the text retrieval system will compute

the following similarity due to the contribution of p37 in both users:

$$sim_C(a, u) = \frac{f_{a,37} * f_{u,37}}{\sqrt{f_{a,37}^2 + f_{a,24}^2} \sqrt{f_{u,37}^2 + f_{u,24}^2}} = \frac{3 * 5}{\sqrt{3^2 + 6^2} \sqrt{5^2 + 4^2}}$$

Using this technique, we can guarantee that the text retrieval system will automatically calculate the positive contributions of the ACS during the simple cosine similarity calculation. But what about the negative contributions due to the cross-products between pMCRs and nMCRs? Lucene’s fields can aid us in solving this problem. Document corresponding to a user can be structured in two separate fields, one to calculate the matches for the positive contributions (pMCR-pMCR and nMCR-nMCR), as already seen, and a second field to calculate the matches for negative contributions (pMCR-nMCR and nMCR-pMCR). The second field will just have the word with prefix ‘n’ replaced with the prefix ‘p’ and vice versa. We refer to the former fields as ‘PRATE’ and the latter field ‘NRATE’. We then execute two queries: one on the field PRATE and the second query on the field NRATE. We merge the result sets from the two queries by subtracting the score from the query on PRATE with the score from NRATE for the same documents, from which we get the following similarity:

$$sim_C(a, u) = \frac{f_{a,37} * f_{u,37} - f_{a,24} * f_{u,24}}{\sqrt{f_{a,37}^2 + f_{a,24}^2} \sqrt{f_{u,37}^2 + f_{u,24}^2}} = \frac{3 * 5 - 6 * 4}{\sqrt{3^2 + 6^2} \sqrt{5^2 + 4^2}}$$

In order to increase the chances of getting at least k results from the merged result set, the depth k of the single kNN queries is increased of a constant expansion factor m . In item-based CF case we proceed in a similar way: the documents take the place of the items and words refer to users.

Now that we obtained that the similarity between documents is identical to that which we obtained using Eq. (1) and (2), we can use the scores obtained by the ACS to predict the missing rate using Eq. (3) and (4), respectively.

4 Experimental Evaluation

Our experiments have been carried out using the public dataset Movielens [8]. Among the three available datasets of Movielens 100K, 1M, and 10M, we decided to use the biggest one, which contains 10,000,054 ratings applied to 10,681 movies by 71,567 users. Starting from the initial data set, five distinct splits of training and test data were generated, the training set (80%) and the test set (20%). The test set acts as a ground-truth for our validation.

To assess the statistical accuracy, we use the mean absolute error (MAE) as the measure of effectiveness, which is given by $MAE = (\sum_{i=1}^N |p_i - r_i|)/N$. Finally, since in our dataset the ratings are quantized, we chose to round the predicted ratings instead of using their real values. Such a process improves the performance of the prediction.

Figure 1 reports the results of user-based and item-based CF approaches for different vales of k . In this evaluation, we compared our approach with the results obtained with a standard approach to CF. In particular, we performed two additional groups of experiments in which the pairwise similarity matrix between

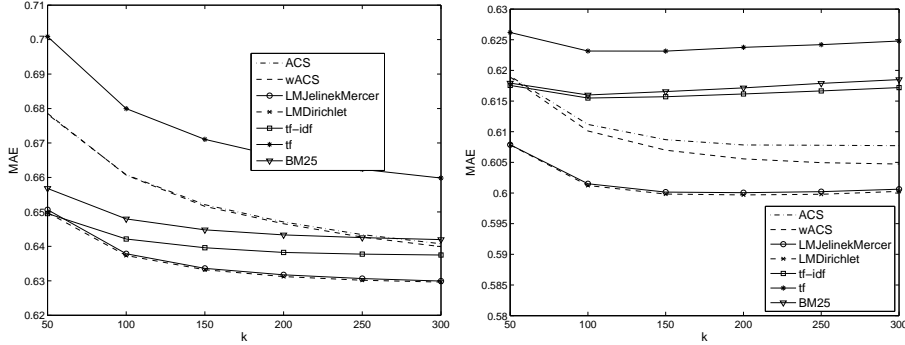


Figure 1: Comparing prediction schemes for user-based approaches (top) and item-based approaches (bottom) and different neighborhood sizes (k).

users (for user-based approach) and between items (for item-based approach) have been computed offline. We used the best-known similarity measure, i.e., ACS. However, since Pearson correlation weighted by Jaccard index has been shown to provide better results than the traditional Pearson coefficients alone [3], we also boosted ACS with Jaccard coefficient. This improves the performance of ACS particularly for the item-based approach. We call this approach wACS. The other five graphs in Figure 1 correspond to the experiment of our approach using Lucene. In particular, tf similarity corresponds to the results obtained with the standard cosine similarity, while $tf-idf$ uses the product of terms frequency and inverse document frequency. We also tried the language models based on the *Dirichlet* and *Jelinek-Mercer* similarities [11], and the *BM25* similarity function derived using the classic probabilistic retrieval model [9]. In all experiments, we used $m = 10$ as query expansion factor. We repeated measures with ten different random partitions of tests/training sets in order to guarantee a confidence level greater than equal to 95%.

According to these experiments, the similarity measures based on Dirichlet and Jelinek-Mercer lead to the best results even versus ACS and wACS correlations on the item-based approach. Moreover, in the user-based approach, MAE improves as k increases. Our solution outperforms state-of-the-art methods currently available (e.g., see [5]).

5 Conclusion

We presented a recommendation framework developed using the Lucene open-source platform. There are numerous advantages to proceeding in this manner, as opposed to developing ad-hoc solutions for CF. To name a few: Lucene provides high performance, robust, and scalable indexing and retrieval platform that is designed to cope with Web-scale data and usage. Moreover, Lucene can be also used to store information of the users and the items over which to

make recommendations. Lucene provides, fielded searching (e.g. title, author, contents). Surprisingly, we found that with our technique the IR language models exhibited a better performance. To accelerate the prediction we use an extra field of float type to store the precomputed average user rates \bar{r}_u or \bar{r}_i . The two Lucene DBs take more or less the same storage space, 86.6MB for the item-based DB and 78.8MB for the user-based DB. The prediction time, including the queries, for both approaches are comparable and about 0.14sec.

References

- [1] A. Bellogín, J. Wang, and P. Castells. Bridging memory-based collaborative filtering and text retrieval. *Information retrieval*, 16(6):697–724, 2013.
- [2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of the 14th conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.
- [3] L. Candillier, F. Meyer, and F. Fessant. Designing specific weighted similarity measures to improve collaborative filtering systems. In *Advances in Data Mining. Medical Applications, E-Commerce, Marketing, and Theoretical Aspects*, volume 5077 of *Lecture Notes in Computer Science*, pages 242–255. Springer Berlin Heidelberg, 2008.
- [4] R. Cöster and M. Svensson. Inverted file search algorithms for collaborative filtering. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’02, pages 246–252. ACM, 2002.
- [5] M. D. Ekstrand, M. Ludwig, J. A. Konstan, and J. T. Riedl. Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys ’11, pages 133–140, New York, NY, USA, 2011. ACM.
- [6] N. N. Liu and Q. Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90. ACM, 2008.
- [7] R. Rafter, M. P. O’Mahony, N. Hurley, and B. Smyth. What have the neighbours ever done for us? a collaborative filtering perspective. In *User Modeling, Adaptation, and Personalization*, volume 5535 of *Lecture Notes in Computer Science*, pages 355–360. Springer Berlin Heidelberg, 2009.
- [8] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

- [9] S. E. Robertson and S. Walker. Okapi/keenbow at trec-8. In *TREC*, volume 8, pages 151–162, 1999.
- [10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of the 10th international conference on WWW*, pages 285–295. ACM, 2001.
- [11] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.
- [12] C.-N. Ziegler. On recommender systems. In *Social Web Artifacts for Boosting Recommenders*, volume 487 of *Studies in Computational Intelligence*, pages 11–20. Springer International Publishing, 2013.